14 Extensive Vocabulary

For this assignment, we would like to study the representative works of two authors and ask ourselves the question "Whose vocabulary is more extensive?"

The authors that we will be looking at are *Charles Dickens* and *Thomas Hardy*. And the representative novels that we will analyze are *A Tale of Two Cities*and *The Return of the Native* respectively. Here are the following steps in our analysis.

Step I: Go to the Project Gutenberg and download the *plain text* versions of both books. Call the first book *Tale.txt* and the second book *Return.txt*. Open the books in a text editor and delete the preamble in the beginning of the books and the license agreement and the closing blurbs at the end of the books. The first book in *Tale.txt* should begin and end with these lines:

```
A TALE OF TWO CITIES

A STORY OF THE FRENCH REVOLUTION

by Charles Dickens

...

it is a far, far better rest that I go to than I have ever known."
```

The second book in *Return.txt* should begin and end with these lines:

```
THE RETURN OF THE NATIVE

by Thomas Hardy

...

kindly received, for the story of his life had become generally known.
```

Step II: The program that you will be writing will be called *Books.py*. The following is the suggested structure. You do not have to adhere to it. However, we will be looking at good documentation, design, and adherence to the coding convention discussed in class. Use meaningful variable names in your program.

```
# Create word dictionary from the comprehensive word list
word_dict = {}
def create_word_dict ():

# Removes punctuation marks from a string
def parseString (st):

# Returns a dictionary of words and their frequencies
def getWordFreq (file):

# Compares the distinct words in two dictionaries
def wordComparison (author1, freq1, author2, freq2):


def main():
  # Create word dictionary from comprehensive word list
  create_word_dict()

  # Enter names of the two books in electronic form
  book1 = input ("Enter name of first book: ")
  book2 = input ("Enter name of second book: ")
```

```
    print()

    # Enter names of the two authors
    author1 = input ("Enter last name of first author: ")
    author2 = input ("Enter last name of second author: ")
    print()

    # Get the frequency of words used by the two authors
    wordFreq1 = getWordFreq (book1)
    wordFreq2 = getWordFreq (book2)

    # Compare the relative frequency of uncommon words used
    # by the two authors
    wordComparison (author1, wordFreq1, author2, wordFreq2)

main()
```

Step III: Declare a global dictionary variable *word_dict*. In the function *create_word_dict()* open the file words.txt and populate the dictionary. Each word in the file will be a key in the dictionary and the value will be 1. Hard code the name of the file, *words.txt*, in your program. When we test your program we will be using a file with the same name.

Step IV: You will have to get the frequency of words in each text to start with. Then here is some additional processing that you will have to do:

- Open the file for reading
- Create an empty dictionary
- Read line by line through the file
- For each line strip the end-of-line character and replace punctuation marks with spaces except apostrophes ('). You should replace all hyphens with spaces. There are hyphens at the end of lines but they do not denote word continuation, so there is no need to check for dangling words on the next line. Remove apostrophes (') at the end of a word or ('s) at the end of a word. If there is an apostrophe followed by a character that is not an 's' keep the apostrophe. In the function *parseString()* create a blank new string. Go through the input string character-by-character. Accept only letters (using isalpha()) and spaces (using isspace()) and apostrophes only for the special cases mentioned above. Return the new string.
- After the dictionary is created close the input file.
- Remove all words that start with a capital letter.
    - Create an empty list for words starting with a capital letter.
    - Go through the words in the frequency dictionary. Check if the word starts with a capital letter.
    - If it does, add it to the list of capital words.
    - Once that list is complete, go through the list of capitalized words and check if the lower case version of that word exists in the frequency dictionary. If it exists, then add the upper case word's frequency to the lower case word's frequency.
    - If the lower case version does not exist in the dictionary, check if it exists in a comprehensive word list that is used for crossword and Scrabble players. If it does, create an entry in the frequency dictionary with the lower case version of the word and the word frequency computed.
    - After you have checked for all capitalized words, remove all those words in the list of capitalized words and their frequencies from the word frequency dictionary.

You should now have a dictionary of words in lower case and their frequencies. You will have removed all proper names of people and places. You will have added the names of people and places that are common words like *Green* and *Bath*. But such words should be few compared to the total number of words that we are dealing with in those novels. You can always write the list of words beginning with a capital letter in a file and examine that file to check if the deletions were properly made.

Step V: In this step you will be working on the function *wordComparison()*. First you will get some statistics of the two novels separately and then you will compare the two together. For each novel compute and print the following pieces of information:

- Print the number of *distinct* words used, i.e. number of words used if you remove the duplicates. Realize, that is just the length of the list of keys for the word frequency dictionary.
- Compute and print the total number of words used by adding all the frequencies together.
- Calculate and print the percentage of distinct words to the total number of words used.

You will create two sets with the list of keys from the two word frequency dictionaries. Let us call these sets *D* and *H* for the two authors respectively. The set difference *D* - *H* represents all the words that Dickens used that Hardy did not. The set difference *H* - *D* represents all the words that Hardy used that Dickens did not. For each of these set differences print the following pieces of information:

- The number of words in that set difference.
- Compute the total frequencies of these words in the set difference (D-H or H-D) and express that as the percentage of total words number in the novel that you found earlier. [In the example below, we computed the sum of the frequencies of the 50 words that Dickens used that Hardy did not (should be 76) and expressed that as a percentage of the 119 words that Dickens used in all.]

Here are two sample files - dickens.txt and hardy.txt taken from the two novels. Your output for these two sample files should be of the following form:

```
Enter name of first book: dickens.txt
Enter name of second book: hardy.txt

Enter last name of first author: Dickens
Enter last name of second author: Hardy

Dickens
Total distinct words = 58
Total words (including duplicates) = 119
Ratio (% of total distinct words to total words) = 48.7394957983

Hardy
Total distinct words = 93
Total words (including duplicates) = 123
Ratio(% of total distinct words to total words) = 75.6097560976

Dickens used 50 words that Hardy did not use.
Relative frequency of words used by Dickens not in common with Hardy = 63.8655462185

Hardy used 85 words that Dickens did not use.
Relative frequency of words used by Hardy not in common with Dickens = 77.2357723577
```

Here is the output of the frequencies of the words in the two excerpts - dickens.out.txt and hardy.out.txt. These outputs have now been checked against the comprehensive word list.